TITLE OF THE INVENTION

OBJECT MODEL MAPPING AND RUNTIME ENGINE FOR EMPLOYING RELATIONAL DATABASE WITH OBJECT ORIENTED SOFTWARE

CROSS REFERENCE TO RELATED APPLICATIONS

A claim of priority is made to U.S. Provisional Patent Application Serial No. 60/069,157, entitled TIER 3 DESIGN SPECIFICATION, filed December 9, 1997 and incorporated herein by reference; and U.S. Provisional Patent Application Serial No. 60/059,939, entitled DATABASE SYSTEM ARCHITECTURE, filed September 26, 1997 and incorporated herein by reference.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

N/A.

BACKGROUND OF THE INVENTION

The present invention is generally related to database technology, and more particularly to interfacing object oriented software applications with relational databases.

The need for interfacing object oriented software applications with relational databases is well known. One method of interfacing an object oriented application with a relational database is to adapt the requests made by the application to the relational database. More particularly, object operations are translated into relational database queries. However, this technique is processor-intensive and sacrifices some of the advantages associated with the object oriented model. As a result, the object oriented software application is unable to function efficiently.

Another method of interfacing an object oriented application with a relational database is to translate database information into a format which is compatible with the object oriented application. Relational databases typically separate data into a plurality of tables through a process known as "normalization" to minimize duplication. A normalized relational database includes a plurality of

ATTORNEY DOCKET NO. ONTO-106XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

tables, wherein each table includes at least one field and one key, and at least one field in each table is uniquely dependent upon the key that is associated with the table. These tables can be translated into objects. However, the objects can become inaccurate when changes are made to the relational database. It is known to adapt to changes in the relational database by performing further translations, but this process requires substantial effort.

BRIEF SUMMARY OF THE INVENTION

In accordance with the present invention, a mapping between an object model and a relational database and a runtime engine are employed to facilitate access to a relational database. The object model can be created from database schema or database schema can be created from the object model. Further, the mapping can be automatically generated. The database schema, object model, and mapping are employed to provide interface objects that are utilized by an object oriented software application to access the relational database.

The present invention provides transparent access to the relational database. The interface objects and runtime engine perform read and write operations on the database, including generation of SQL code. Consequently, neither programmers nor software applications need have knowledge of the database structure, the database programming interface, database security, or the database transaction model in order to obtain access to the relational database. Further, changes to the relational database do not always necessitate additional mapping.

BRIEF DESCRIPTION OF THE DRAWING

Other features and advantages of the present invention will become apparent in light of the following detailed description of the drawing, in conjunction with the drawing, of which:

Fig. 1 is a block diagram that illustrates use of the

ATTORNEY DOCKET NO. ONTO-106XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

map to generate interface objects that are employed by a runtime engine and an object oriented software application to access a relational database;

Fig. 2 is a block diagram of database schema;

Fig. 3 is a block diagram of an object model;

Fig. 4 is an object diagram of a mapping;

Fig. 5 is an object diagram of the runtime engine;

Fig. 6 is an object diagram of RtCore.DLL; and

Fig. 7 is a sequence diagram that illustrates operation of the runtime engine.

DETAILED DESCRIPTION OF THE INVENTION

Referring to Fig. 1, a mapping tool 10 is employed to generate a map 12 in which relationships between an object model 14 and schema associated with a relational database 16 are defined. A code generator 18 is employed to examine the relationships that are defined in the map 12 and a model object oriented interface associated with an object oriented software application 22 to generate interface objects 20. The interface objects 20 are employed by the object oriented software application 22 to access the relational database 16 via a runtime engine 24, which also uses the map 12 to drive its processing.

The object model 14 is a template that has a predetermined standardized structure. The illustrated object model includes attributes and inheritance relationships that are mapped to relational database features such as tables, rows, columns, keys, and foreign keys. Mapping the object model to the relational database schema includes mapping a class attribute to a table column, mapping a class attribute to a 1-1, 1-N, or N-N relationship, and mapping class inheritance to rows within a table or across tables.

Referring now to Figs. 2, 3 and 4, the mapping of a class attribute to a table column can be described generally as: Class Attribute -> Table Column + Class Key + Joins. Mapping the class attribute defines where the attributes are read from and written to. In the illustrated example, the

ATTORNEY DOCKET NO. ONTO-106XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

class attribute CPerson.name 26 maps to table column TPerson.name 28. The "Class Key" is employed to relate an object instance to a row in the table. In particular, key values of the class are mapped to columns in a table that comprise the primary key. In the illustrated example, CPerson.id 30 maps to TPerson.id 32. "Joins" defines keys between tables within a class. Since there is only one table in the mapping of Cperson.name to Tperson.name, no information is required for Joins. If Cperson includes two tables, such as Tperson and X, then mapping Cperson.y to X.y includes: Cperson.y Maps to X.y + Cperson.id Keys to Tperson.id + Tperson.id Joins to X.id.

Mapping a class attribute to a 1-1, 1-N, or N-N relationship with at least one other object can be described generally as: Class Attribute -> Class + Class Attribute -> Foreign Key + Joins. When an object has associations to other objects, an attribute in the object points to one or multiple other objects. If the object points to only one object, there is a one-to-one (1-1) association between the objects. If an object points to multiple objects, there is either a one-to-many (1-N) or many-to-many (N-N) relationship between the objects. In the illustrated example, CEmployee.dept 34 maps to Cdepartment 36, where CEmployee 38 to CDepartment 36 is a one to one relationship. "Foreign Key" represents identifying the foreign key. If CEmployee is related to CDepartment, there is a foreign key to another table. The foreign key is identified within one of the tables that comprise Cemployee and is related to the class attribute Cemployee.dept. This relationship may be inferred from foreign key information in the database schema. It is also possible that foreign key information is missing or that there are many foreign keys in CEmployee to CDept. Consequently, this step involves selecting columns that represent the foreign key. In the illustrated example, Cemployee.dept is associated with TEmployee.deptid. Once a class attribute is associated with the foreign key which resides in that class, "Joins" is defined to associated

ATTORNEY DOCKET NO. ONTO-106XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

classes. In the illustrated example, TEmployee.deptid joins to TDepartment.id is defined.

Mapping class inheritance to rows within a table or across tables is performed by specifying a WHERE clause on the class which can distinguish the class from the associated parent class. This information is stored in the mapping model.

Table 1 describes how an object model can be mapped to structures in a database schema.

ATTORNEY DOCKET NO. ONTO-106XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

Table 1

In object model:	Can be mapped to:
A single class	All or selected columns in a table. A WHERE clause can be associated with a class to specify which rows of the table belong to the class. Multiple tables that are joined by the same primary key, or by a unique foreign key relationship. If the same data is stored in multiple tables, the duplicate columns can be handled by mapping one of the table.columns for read, and all of the columns for insert and update. Multiple tables, possibly in different databases, which have similar column definitions (e.g., EastEmployees and WestEmployees tables can be merged as a single Employees class). Multiple tables that are unrelated in the database, if a logical relationship can be defined in the mapping.
Single inherited classes	A single denormalized table. Columns that contain data for all records are typically mapped to the superclass, and a WHERE clause is defined for each subclass as a discriminator for selecting which rows belong to the given subclass. Multiple tables that have the same primary keys. To ensure uniqueness of records, the primary keys of "subclass" tables may also be defined as foreign keys for the "superclass" table.
Multiple inherited classes	A single denormalized table. Different columns are mapped to each of the classes. Typically, there are multiple columns that can be used as indexes. The subclass mapping has multiple joins defined, which are used to traverse each of the inheritance relationships. Multiple indexed tables. The table that is mapped to the subclass has multiple keys corresponding to each of the keys in the superclass tables. To ensure uniqueness of records in the subclass table, the key on the subclass table may not be defined as a multi-column key. If all the data of the superclass tables is duplicated in the subclass table, then no join is required to instantiate an instance of the subclass. However, joins would be needed to ensure integrity of data when performing insert, remove, and update operations.

ATTORNEY DOCKET NO. ONTO-106XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

Object relationships are mapped to the database schema by defining the joins needed to access related objects or groups of objects (lists). The joins make use of foreign keys defined in the tables that are mapped to the related classes. Table 2 describes mapping of object relationships relative to the illustrated example.

ATTORNEY DOCKET NO. ONTO-106XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

Table 2

Object Relationship:	Mapping Example:
1-1 object relationship	Department can have only one Manager. Department class has an object attribute (Manager) that references only one instance of the Employee class. Department table has a foreign key column (ManagerID) that references only one row in the Employee table. A join is defined for the Department class based on a unique foreign key.
N-1 object relationship	Many Employees have one Department. The Employee class has an object attribute (Department) that references one instance of the Department class. The Employee table has a foreign key column (DeptID) that references one row in the Department table. A join is defined for the employee class to access the one row in the Department table that is referenced by the foreign key in the Employee table.
1-N object relationship	A Department has many Employees. The Department class has a list attribute (Employees) that references the related group of employees. Employee table has a non-unique foreign key that references the Department table. A join is defined for the Department class that selects all rows in Employee that matches the current Department's Deptid.
N-N object relationship	An Employee can have many Projects. A project can have many Employees. The Employee class has a list attribute that references a group of Projects, and the Project class has a list attribute that references a group of Employees. This mapping uses joins based on the join table that relates the Employee table and the Project table. The Employee class uses a join to select rows from the Project table that match the current instance's Employeeid. The project class uses a join to select rows from the Employee table that match the current instance's Projectid.

Table 3 describes how structures in a database schema can be mapped to structures in an object model.

Table 3

In a database schema:	Can be mapped to:
Rows, discriminated by WHERE clause	All attributes of a single class
A single table	All attributes of a class, assuming the other persistent attributes of the class are mapped to columns in other tables. Multiple classes (effectively, a vertical split of the table) A single-inherited classes (if at least one column is appropriate for discriminating selection of rows for subclasses) Multiple-inherited classes (if the table has multiple indexes)
Multiple tables, different columns	A single class (if the key structure exists to join row uniquely) Multiple classes (unrelated, unless key structure exists to support joins)
Multiple table, same columns	A single class that represents a logical merge of the tables. (NOTE: Primary key values must be unique between the tables.)
Multiple tables, same primary key	Single inherited classes (each table represents a class, keys used to define joins between subclass tables and superclass table. Single class (with joins based on primary key)

If the same data is stored in multiple tables, the duplicate columns can be handled by mapping one of the table.columns for read, and all of the columns for insert and update.

Schema relationships are mapped directly to object relationships, either in the form of object attributes or list attributes. In general, a foreign key in the database schema is mapped to an inverse relationship between an object attribute (on the class mapped to the table holding a foreign key) and a list attribute (on the class mapped to the table referenced by a foreign key). A join table is mapped to an inverse relationship between list attributes defined on each of the classes mapped to the tables related by the join table.

ATTORNEY DOCKET NO. ONTO-106XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

Table 4 describes how relational keys are mapped to object relationships relative to the illustrated example.

Table 4

Schema Relationship	Corresponding Object Relationship
Unique Foreign Key	1-only-1 object relationship represented by an object attribute with a cardinality of one on the class mapped to the table that has the foreign key. This relationship can also be mapped as an embedded type.
Non-Unique Foreign Key	N-1 object relationship, represented by an object attribute on the class mapped to the table with the foreign key, and a list attribute on the class mapped to the table referenced by the foreign key.
Join Table (with no other data columns)	N-N object relationship, represented by a list attribute on each of the classes mapped to the tables related by the join. Each list attribute represents a collection of references to objects of the other type.
Join Table (with additional data columns)	A class mapped to the join table, AND a N-N object relationship, represented by a list attribute on each of the classes mapped to the tables related by the join.

Referring to Figs. 5 and 6, the runtime engine comprises a plurality of dynamic link libraries ("DLLs") including: RtMap.dll 50, RtCore.dll 52, DbObjs.dll 54, OiGenBase.dll 57, and a set of generated DLLs 56. The generated DLLs 56 contain one COM interface and implementation class for each class defined by a mapping model. A mapping model binary file is generated in parallel with each DLL containing the mapping information associated with the DLL. The RtMap.dll 50 implements the classes that can load the information from the binary file at runtime and make it available to the runtime interface objects associated with DLLs 56 and to the client objects 58 of the generated COM objects through a set of predefined COM interfaces.

Classes OOBase and OObject in RtCore.dll 52 form the core of the runtime engine 24. The OOBase is a base abstract class which is used as a base for all the generated implementation classes. The generated classes are ATL COM objects implementing one of the standard IDslObject/IDslList/IDslQlist and one or more of the client

interfaces (e.g., Employee). The ATL implementation classes have state implemented as a set of attributes of the primitive types called the "front state" (or the front data set). The OOBase contains a pointer to the OObject and a public pure virtual method to access the address of each attribute in the classes descending from itself. The attributes are indexed according to the class definition for the object. The OObject class is abstracting the runtime functionality for a generic object. It contains a set of attribute info-value pairs (one per attribute, constructed when the object is initialized to form a "back state," or baseline). OObject also has a set of attribute flags (one per attribute, bitwise or of values like isModified, isRetrieved, isDirty, isNull and others). One instance of the OObject is created for every instance of the generated objects to take care of the interface to the persistent data storage through a set of DB objects that are MTS stateless, transactional objects.

Fig. 7 illustrates the sequence of actions that take place when a business object creates a Dsl object in step 61, accesses the name property in step 65 and saves the object in step 69. OObject is constructed when the constructor of the DPerson (the generated COM implementation class) is invoked in step 62. The constructor passes as parameters the appropriate constant ClassInfo reference and a reference to itself. The OOBject initializes all flags and sets the attributes to the default values as defined in the AttrInfo objects associated with the ClassInfo in step 63. The GetAttrPtr() function defined by the OOBase is employed to get the attribute address for each attribute in the class in order to initialize the front set of attributes on the object in step 64.

When the getName (propget) of the generated object is called in step 65, the generated code checks to see if the attribute was retrieved. If the attribute was retrieved then the cached value is returned. Otherwise, RetrieveAttrValue() of the OObject is called in step 66, passing the id of the

ATTORNEY DOCKET NO. ONTO-106XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

desired attribute (name in the example). The OObject will look at the fetch matrix for this attribute and see what other attributes should be retrieved with it in step 67 and then determines what tables and columns are involved, how they are joined and executes the appropriate SQL statements using the stateless MTS object. The GetAttrPtr() function defined by the OOBase is employed to get the attribute address for each attribute in the class in step 68.

When the object is saved in step 69, the generated code calls the OObject SaveChanges() method in step 70. The OObject determines what attributes have changed and, depending on the concurrency control mode in effect, makes sure the appropriate locks and transactions are set and respectively open and then executes the appropriate SQL to write the data to the persistent storage in step 71.

Referring again to Figs. 1-4, the runtime engine also includes a plurality of performance enhancing features such as optimized data retrieval algorithms. An attribute retrieval can be associated with each attribute to optimize attribute retrieval from the database. As a default case, all attributes are retrieved when any one of an object's attributes are needed. However, the attribute retrieval list for any attribute can be edited to specify different attribute retrieval behavior. For example, a request for an Employee Id may cause the Photo attribute to be dropped from the attribute retrieval list on the Id attribute if that data resides in another table and is only infrequently used. Attribute retrieval lists are a performance feature that enable optimized data access by only doing JOINS and additional SELECT statements when the data returned by those actions is needed.

Performance is also enhanced by "just in time" data retrieval. By default, whenever an attribute value is read from the database, all of the other attributes for that instance are also read. However, Data Component Developers are permitted to modify the mapping information for a Data Component to define an attribute retrieval group for each

ATTORNEY DOCKET NO. ONTO-106XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313

attribute of a class that determines which other attribute values are returned when the requested attribute is read from the database. This makes it possible to avoid executing JOINs or SELECTs to retrieve data that may not be needed. For example, assume that a class, CPerson, has four attributes: Id, Name, Zip, and Photo, and the Photo attribute is mapped to a column in a different table from the others. The Data Component Developer may drop Photo from the group of attributes that are retrieved when either Id, Name, or Zip are read. A query is issued to get the Name and Id of a instance of CPerson where Id = 10. Based on the attribute retrieval information, the run time engine retrieves only the values for the person.id, person.name, and person.zip attributes, thus avoiding an unnecessary join to return the photo attribute value as well.

If an object does not have an attribute in memory when an attempt is made to use that attribute, the object will issue a SELECT statement to retrieve the attribute from the database. "Just-in-time" attribute population allows the object to be populated with the minimal amount of information necessary for the application while still making any remaining information available when it is needed.

Lazy reads are also employed to enhance runtime performance. When a query is defined to identify objects for retrieval from the database, the SQL SELECT statement is not issued immediately. Queries are executed only after an attempt has been made to use or modify the resulting data.

Having described the embodiments consistent with the present invention, other embodiments and variations consistent with the present invention will be apparent to those skilled in the art. Therefore, the invention should not be viewed as limited to the disclosed embodiments but rather should be viewed as limited only by the spirit and scope of the appended claims.

ATTORNEY DOCKET NO. ONTO-106XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & HAYES LLP
TEL (617) 542-2290
FAX (617) 451-0313